OMID 2019 Team Description

Ali Mollajafai[1], Mohammad hossein Zahedi[1], Alireza Fatollahzade[1], Rasoul Aboutalebi[1], Alireza Sahebi[2], Javad Rahmani[5], Omid Mahdizadeh[6], Yashar mahmoudi[1], Hashem Khan mohammadi[1], Mohammad hossein Zolfaghari[3], Maryam Akhoundian[4]

[1] Department of Electrical Engineering of Shahed University of Tehran, Iran

[2] Department of Computer Engineering of Shahed University of Tehran, Iran

[3] Department of Biomedical Engineering of Shahed University of Tehran, Iran

[4] Department of Computer Science of Shahed University of Tehran, Iran

[5] Department of Electrical Engineering Islamic Azad University Science and Research Branch

[6] Department of Electrical Engineering of Khaje Nasir University of Tehran, Iran

http://www.omidrobotics.ir

omid.robotics.ssl@gmail.com

Abstract. This paper is an explanation of OMID 2019, Robocop small size team, technical estate and robots technical improvement which generally divided into three part: Mechanical part, Electronic part and Software part.

1. Introduction

Omid Robotics Team (ORT) began small size team in 2007. ORT has been participated in competitions since 2007 as a branch of robotics society of Department of Electrical Engineering of Shahed University. The team is located in Tehran, Islamic Republic of Iran. The previous work of the team was used as reference.

We are willing to invite talented students. In addition, newcomers are mentored by former members.

This article describes the team's general information and improvements in this year. The designed robot systems of ORT consist of three major parts: mechanics, electronics and software.

ROBOTICS TEAM

2. Mechanical Design

According to Small-Size League rules, the robots must have specific dimensions. Our robots have a diameter of 178mm and height of 148mm and each robot covers less than 20% of ball. The whole robot is about 2 kilograms weight. 3D simulation models shown in Fig. 1 are created with Solid Works.

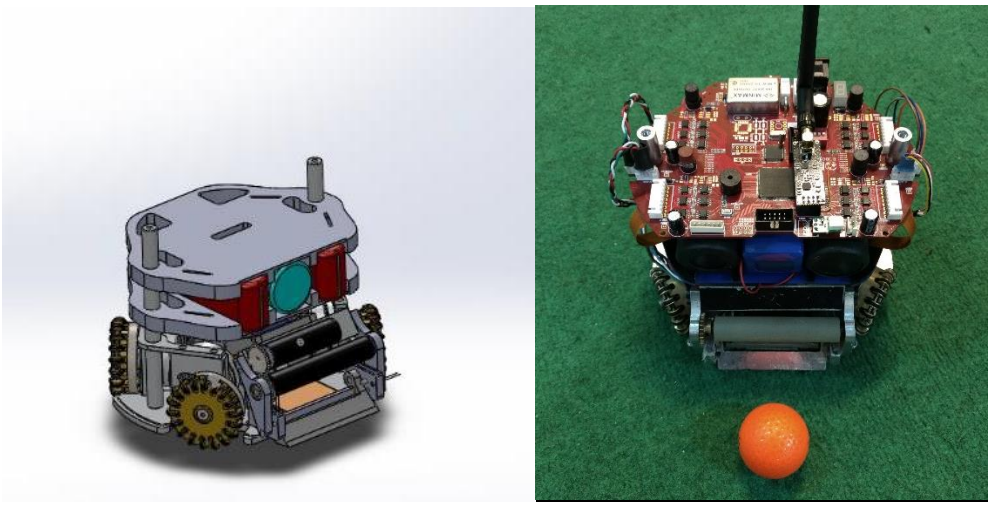

Fig. 1. Robot's mechanical plan design

3. Electrical System

Robots consist of three general parts: Communication, Main board and Kick board. In the main board we use FPGA Xilinx Spartan 6 XC6SLX9-2TQG144I [1], STM32F103 ARM processor, motor drivers, etc. A 4-cell li-polymer battery (1500 mAh) used as the power supply. The block diagram of robot's electrical hardware is shown in Fig.3.

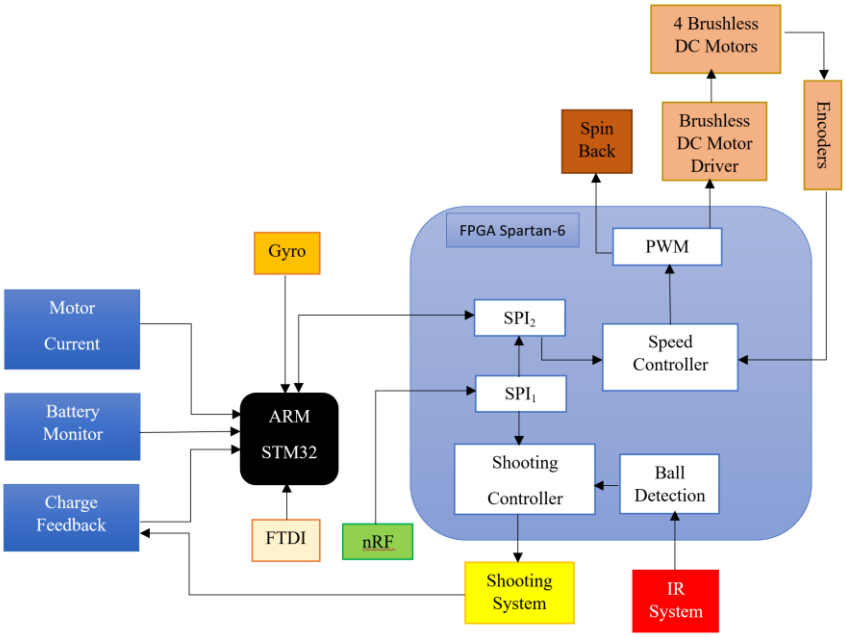

Fig. 2. The Block Diagram of Robots Hardware

4. Software

This year our improvements in software are divided into three parts:

4.1. Motion Control

In our last TDP [6] we discussed about PID as a controller of the robot's motion. We had some problems with this controller, for example we wanted the robot to move at the highest speed until near the destination, and then the PID controller to be used, so that the robot can reach the destination as fast as possible, but equalizing the highest speed and the first generated speed of PID for the robot was difficult for us.

Therefore, we decided to create a unified speed generator system instead of a two-part system, called the "Speed Diagram". This system receives the position of the robot and the destination as an input and generates the speed that the robot must have as an output. It can be shown as below:

$$\mathbf{MSp} - [((\mathbf{MEr} - \mathbf{e}) / \mathbf{d})\text{^}\mathbf{p}]$$

Where:

- e is the error between the current position of the robot and the destination (e = destination - current position)
- MSp is the maximum speed that robot can go
- MEr is the minimum error (distance to destination) that the robot should move with speed MSp
- p is a value that determines the slope of the diagram that can be used to determine whether it's important reach the destination carefully or quickly is more important.
- d can be achieved by following equation:

$$\mathbf{d = MEr / (MSp\text{^}(1 / p))}$$

Therefore, we have three parameters that needed to be set: MSp, MEr and p. The effect of changing these values is shown below:

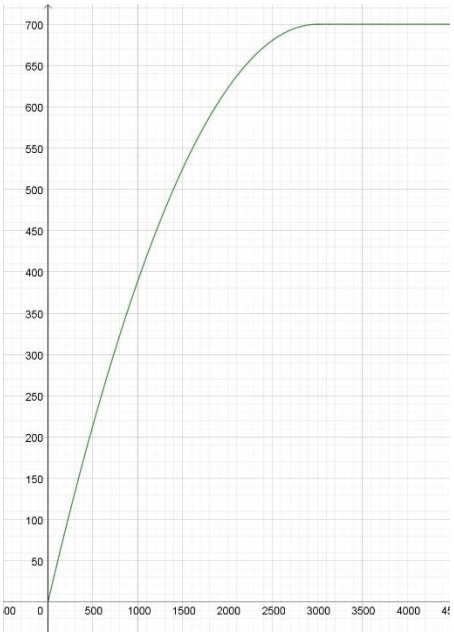

Fig. 3. MSp = 700, MEr = 300, p=2

Reach the destination quickly

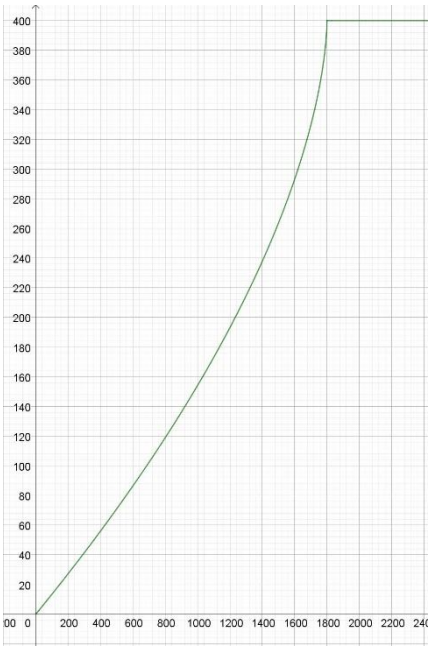

Fig. 4. MSp = 400, MEr = 1800, p=0.6

Reach the destination carefully

4.2. Path Finding

As described in previous TDP [6], we use RRT [7] as a path finding algorithm. This year we made some improvements in our implementation that one of them will be described below:

Finding a path to inaccessible destination:

If a robot wants to go to a destination and that destination is surrounded with balks that cannot be crossed, we need to find a place outside that surrounded area, the nearest to the robot's position. To do this, we need to consider a circle with the destination as the center and the robot's radius (or any other value) as the radius. Then we need to get the points on the circle which have the same angular distance to each other. After that we must check if we can find the path to the nearest of them, if we could, then the process is done, if not, it is obvious that the circle is in surrounded area, so we need to increase the radius of circle and try the above process again.

We should set two parameters, one is "Radius increase rate", the value that will be added each time to the circle's radius, the other is "Angular distance between points", the angular distance between points on the circle. Reducing "radius increase rate" and increasing "angular distance between points", will increase accuracy of finding the nearest position. Images below show the effect of changing these parameters on the selected point as destination. The blue robots are balks, the yellow robot is one that want to go to the destination, the red dot in the center of circles is the primary destination and the red dots on circles are candidates to be the final destination.

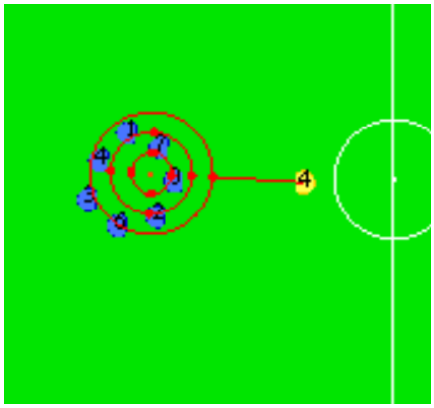

Fig. 5. Radius increase rate: 2*robot's radius
Angular distance between points: $\pi/2$

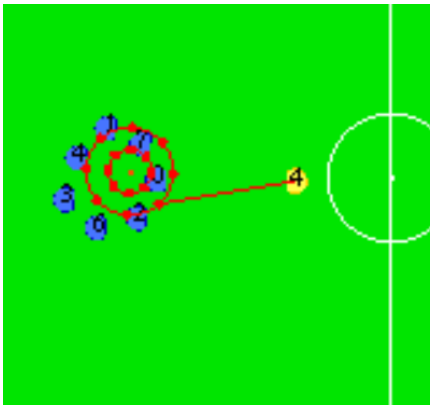

Fig. 6. Radius increase rate: 2*robot's radius
Angular distance between points: $\pi/4$

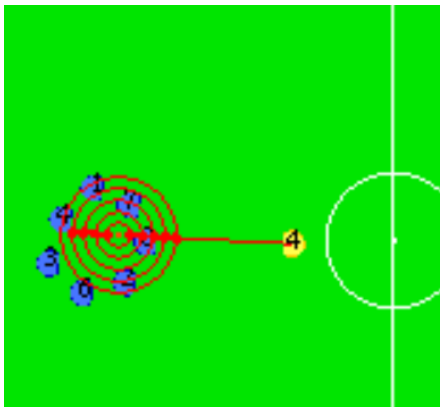

Fig. 7. Radius increase rate: robot's radius
Angular distance between points: π

4.3. Decision Layer

As described in the previous TDP [6], Some functions in AI code are specified to help the robot to do it's best depending on which situation is provided by the position of the robot, other teammates and opponents. Also, it is important where the ball is and which robot controls it. By these functions, the robot decides which duties how can be done in this situation.

This year, we improved these functions, Moreover for improvement of the robot's decision, make it more intelligent depending on the robot's terms (like position and ball owning) and game terms (like stop or play). For each step, each robot (just in our team) should take place as a defender or attacker.

We defined a state which shows the mode of the game (on the play or not) and action which shows the operation that the robot can take, related to the represented state. To perform it, we weighted actions from one state to another to decide which one is appropriate in this step. Then all of these shape the game in the best way. Every robot should choose an action in a way that is good for other teammates, in other words, each robot chooses the best action for itself and also helps other teammates to choose their best. In addition, the weight of actions for each robot depends on other robot's situation and decision.

It should be mentioned that the process above is only implemented for one of the robots, but it will be completed soon for the others. So we just mentioned what it would be in the collaborating environment and did not explain it more.

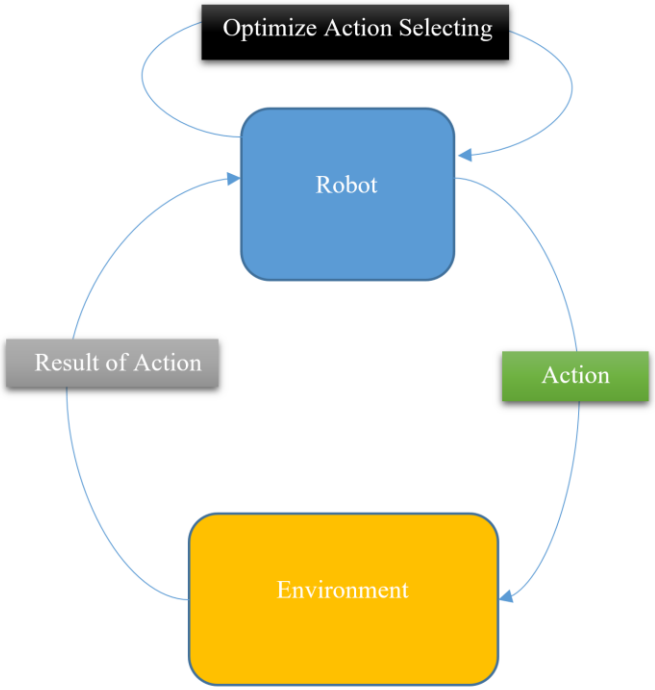

Fig. 8. Decision making procedure for selecting the duty of the robot per step

In this chart, optimize action selecting refers to learning process. In this process, the robot learns if this action is good for this state according to the result of action. Then keep it in it's memory to use it for forward decisions.

Reference

1. Xilinx : Spartan-6 FPGA Family data sheet
2. Li Wang, Zheng Zhang, Ping Sun: Quaternion-Based Kalman Filter for AHRS Using an Adaptive-Step Gradient Descent Algorithm
3. Sebastian THRUN, Wolfram BURGARD, Dieter FOX. : PROBABILISTIC ROBOTICS.
4. Kanjanapan Sukvichai, Piyamate Wasuntapichaikul, Yodyium Tipsuwan. : IMPLEMENTATION OF TORQUE CONTROLLER FOR BRUSHLESS MOTORS ON THE OMNI-DIRECTIONAL WHEELED MOBILE ROBOT
5. Li-Chun Lai, Chia-Nan Ko, Tsong-Li Lee, Chia-Ju Wu. : Time-Optimal Control of an Omni-Directional Mobile Robot.
6. Omid Robotics Team. : OMID 2018 Team Description paper. Technical report, ECE Department, Shahed University of Tehran.
7. LaValle, Steven M: "Rapidly-exploring random trees: A new tool for path planning". Technical Report. Computer Science Department, Iowa State University.